Program E1FILTER *4*
About this document **4**
Author **4**
Briefly about E1FILTER *4*
Short description **4**
Errors **4**
Limitations **4**
Version numbering **4**
Command format *4*
FVALUE-commands **5**
Phase "Initialize the program" *6*
MaxColCount=Maximum number of columns in input- and output-file **6**
MaxScanCount=Maximum number of selection conditions **6**
NumEdit=Item edit for formatting numeric output values in the evaluator **6**
ItemSep=Item separator for list-handling in the evaluator **6**
Phase "Select input file(s)" *6*
InFile=name of the input-file **6**
fmk.age, fmk.size, fmk.name and fmk.contains **6**
FMK.AGE=days - selection by the age of the file **7**
FMK.SIZE=min-max, selection by the size of the file in bytes **7**
FMK.NAME=namestring, selection by the name of the file **7**
FMK.CONTAINS= string1,sting2,string3,…, selection by the content of the file **8**
Phase "Read input file block" *8*
InLineMask=list of binary fields/block read descriptions **8**
LineReadCmd=Block read description **9**
MaxLineLen=Maximum length of the input line **9**
LineEdit=Line edit formatting instructions **9**
ConvFile=name of the character conversion file **9**
Phase "Build input line" *10*
Inform=Input-file format **10**
- INFORM=XML, eXtensive Markup Language 10
- INFORM=STRUCTURED, structured format export file 11
Parameter FILTERID.<field name>.<column number>=column number 11
InSep=column separator of the input-file **12**
- Insep=NONE 12
CopyPrevCols=Copy Previous Column values **12**
InHeaderLen=number of the lines in the header **12**
HeaderFile=name of separate Header file (column title file) **12**

Phase "Start/stop reading input file" *13*
StartReadForm=Condition formula for starting the reading **13**
StopReadForm Condition formula for stopping the reading **13**
Phase "Select input lines" *13*
Incln, where n=1-9, line selection conditions (Include) **13**
Excln, where n=1-9, line exclusion conditions (Exclude) **13**
Phase "Build output line" *13*
Oc=Output columns **13**
- Oc, additional parameters 14
- Oc, formatting and header text 14
- Oc, summarizing function 14
- Oc, giving long list of consecutive integers 14

OutSep=column separator of the output-file **14**
RepSep=Separator substituting OutSep-character in the input-item **14**
ItemEdit=Item edit formatting instructions **14**
Phase "Select output lines" *15*
PrintOutCond=Condition formula to print out the line **15**
Phase "Sort" *15*
SortKeyn=Sorting Parameters number n, n=1 to 9 ..... **15**
Phase "Allocate values to calendar days" *15*
General logic of Calendar allocation **15**
PeriodForm=formula returning the time period **16**
PeriodCol=number of column containing the time period **16**
CalDateFormat=date format for calendar day output **16**
SplitCols=list of columns to be split into daily shares **17**
ExcludeFile=name of the file containing dates to be excluded **17**
IgnoreExcludeFileForm=Boolean formula: ExcludeFile should be ignored **17**
ObjectForm=Formula returning the object (string) **17**
ExclusiveForm=Is an exclusive allocation (reservation) **17**
TimeScope=Time period limitation for calendar allocation **17**
SplitValuesForm=Formula telling, if split values are copied 1:1 to dates **17**
MaxAllocCountForm=Formula returning maximum number of allocated days **18**
Phase "Summarize" *18*
SumKeyForm=Summary key calculation formula **18**
Summary function (in parameter oc) **18**
Phase "Create output columns" *19*
OutFile=name of the output-file **19**
Append=Append to the output file **19**

HeaderOut=Y/N **19**
OutLineEndStr=character string **19**
Phase "Format output" ***20***
LineTemplate=Template for line output **20**
HeaderTemplate=Template for header block output **20**
FooterTemplate=Template for footer block output **20**
Application functions of E1FILTER ***20***
Hints, tricks, warnings and examples ***21***
Using filenames longer then eight characters **21**
Using full path names **21**
Creating a log-file **21**
Displaying the results **21**
Using Fvalue-values in the command-line **21**

Program E1FILTER

About this document

This document is the description and users manual of program E1FILTER.

Author

Program E1FILTER is written by Eino Uikkanen (mail: eino.uikkanen@iki.fi) for free use of anyone.

Briefly about E1FILTER

Short description

Program E1FILTER manipulates and filters binary and text-files with column-based structure. Column lengths can be variable or fixed. E1FILTER is designed to be used mainly as interface between different systems, by filtering, converting, formatting and allocating the input data to other formats.

E1FILTER is a DOS program, but runs perfectly under Windows-9x, -NT, -2000, XP etc.

E1FILTER if designed to be used in batch mode only. Therefore the start-up of E1FILTER-usage may require some effort especially of a person, who normally uses only interactive programs.

E1FILTER is utilizes formula evaluator module described in separate document Evaluator.doc.

E1FILTER parameters and application functions are listed also in Excel-file E1FILTER.XLS,

Errors

I have tried to minimize the amount of programming errors in E1FILTER by building it on carefully designed and tested program primitives.

However, I don't guarantee anything and I don't take any responsibility of correctness of E1FILTER's functioning or results.

Limitations

Because E1FILTER is a DOS-program, it doesn't understand file- or pathnames longer than 8 characters. This restriction can however be easily overridden as explained in section "Hints, tricks, warnings and examples".

Version numbering

E1FILTER has no version number, but you can check the version date by command E1FILTER /?.

Command format

The command format of the program E1FILTER is:

E1FILTER [input-file [output-file]] [Parameters]

Name of the input file can be given as the first command line parameter or as a command parameter (InFile=).

Name of the output-file can be given as the second command line parameter or as a command parameter (OutFile=).

Command Parameters can be given in the command line or be set with the program FVALUE prior to calling E1FILTER. These two methods can be mixed. If same parameter is given both using FVALUE mechanism and in the command line, command line value is used. Below some examples of setting the program Parameters with program FVALUE.

```
fvalue E1FILTER.Oc=18,5,6,11,4,8,33
fvalue E1FILTER.InSep=@9
fvalue E1FILTER.OutSep=@9
```

Examples of calling the program E1FILTER

```
E1FILTER d:\invoice\september.txt
E1FILTER
```

FVALUE-commands

This section is a brief description of program FVALUE.EXE. You find more detailed and comprehensive description from document FVALUEGB.DOC.

You can set the value of a E1FILTER's parameter by command *fvalue E1FILTER.<parameter name>=<parameter value>* e.g.:

```
fvalue E1FILTER.INFILE=C:\INVOICE\SEP2005.TXT
```

You can initialize the value of a E1FILTER's setting the value to blank by command *fvalue E1FILTER.<parameter name>=* (N/B equal sign after parameter name) e.g.:

```
fvalue E1FILTER.linedit=
```

You can check the value of a E1FILTER's parameter by command *fvalue E1FILTER.<parameter name>*, e.g.:

```
fvalue E1FILTER.oc
```

You can see the values of all parameters set for E1FILTER by command *fvalue E1FILTER.* (N/B dot after E1FILTER), e.g.:

```
fvalue E1FILTER.
```

Program FVALUE writes the parameters in a file, one parameter and value per line in the format *Parameter=value*, e.g.

```
InHeaderLen=1
insep=,
outfile=c:\resports\invoice.txt
outsep=@9
```

This is the standard FVALUE-format; two columns separated by equal sign - the name of the parameter in the first column and the value of the parameter in the second column. This definition can be overridden; the column separator and the column numbers can be defined differently. This is done by giving the number of the input-column (which contains the parameter name), the column separator and the number of the output column (which contains the parameter value) after the parameter name in format *Parameter name/input column number/column separator/output column number*, e.g.

Advanced format; user defined input column number, separator and output column number:

```
Set    FVALUE COLORS.BACK/3/@9/5=5
Remove FVALUE COLORS.BACK/3/@9/5=
Show   FVALUE COLORS.BACK/3/@9/5
```

Standard format; input column number=1, separator="=" and output column number=2:

```
Set    FVALUE COLORS.BACK=5
Remove FVALUE COLORS.BACK=
Show   FVALUE COLORS.BACK
```

Phase "Initialize the program"

Parameters *MaxScanCount* and *MaxColCount* define maximum number of filtering selections (parameters *incl* and *excl*) and number of columns in one line. You need to set these parameters if and only if you need higher values than the default values 10 filtering selections and 200 columns per line.

MaxColCount=Maximum number of columns in input- and output-file

By default the input- and output-files can have maximum 200 columns. If either of the files has more columns, set the needed number of columns with this parameter. Don't set the number of the columns unnecessary high, because it increases also memory usage even when the actual number of columns read is less than the maximum number.

MaxScanCount=Maximum number of selection conditions

By default the total number of selection conditions in including and excluding selection conditions (Incl1-Incl4 and Excl1-Excl4) is 10. This can be changed with the parameter MaxScanCount=.

Don't set the maximum number of the selection conditions unnecessary high, because it increases also memory usage even tough the actual number of selection conditions is less than the maximum number.

NumEdit=Item edit for formatting numeric output values in the evaluator

A formatting instruction, which is applied to every numeric output in the evaluator. In this function O$ represents the edited value.

Default value: no formatting

ItemSep=Item separator for list-handling in the evaluator

A formatting instruction, which is applied to every numeric output in the evaluator. In this function O$ represents the edited value. Characters in the parameter-string can be given in form of @nnn, where nnn is character code

Default value: character 3

NB: In evaluators test program ES.EXE *Item separator* is comma (,)

Phase "Select input file(s)"

Input file name is given in parameter *infile*, where you can use wildchars ? and *.

If *infile* refers to a set of files instead of one individual file, you can also use general FMK-parameters *fmk.age, fmk.size, fmk.name* and *fmk.contains* to select files based on the age, size, name and content of the file.

InFile=name of the input-file

Wildchars ? and * can be used in the name of the input-file.

fmk.age, fmk.size, fmk.name and fmk.contains

If you refer to a group of files by using characters *or ? in the name of the input-file, you may want to use file selection parameters. With file selection parameters you can select converted files based on files name, age, content, size etc.

Because file selection Parameters are used in many programs, the naming of these parameter differs from the naming of the E1FILTER-specific parameters. In file selection parameters 'E1FILTER' must be substituted by 'FMK' (File Management Kit).

`E1FILTER.insep=@9`	Setting E1FILTER-specific parameter
`Fmk.age=30`	Setting file selection parameter (age<=30 days)

In the command line the parameter is always given without the "group-name", E1FILTER or fmk.

```
E1FILTER infile=*.pcd size=-10000 age=14
```

File selection parameters are not initialized in the batch run FilteIni.bat. You can initialize the file selection parameters with the command fvalue fmk.*=. If you use file selection parameters, it is recommended to give the initialization command also at the end of the batch run.

`fvalue fmk.*=`	Initialize file selection parameters

FMK.AGE=days - selection by the age of the file

Checks the time stamp (created or modified) of the file and selects the file if the age of the file is less than or equal to age (days).

`Fvalue fmk.age=30`	Select, if the age of the file is less than or equal to 30 days
`Fvalue fmk.age=0`	Select, if created or modified today

Selected time range can also be given as date-range in format dd.mm.yyyy-dd.mm.yyyy

`Fvalue fmk.age=1.9.2004-`	File is created or modified 1.9.2004 or later

FMK.SIZE=min-max, selection by the size of the file in bytes

Selects the file, if the size of the file in bytes is between the given limits.

`size=-10000`	Size is less than or equal to 10000 bytes
`size=2000-`	Size is at least 2000 bytes
`size=2000`	Size is at least 2000 bytes (without dash)

FMK.NAME=namestring, selection by the name of the file

Selects the file, if the namestring matches with the file-name. Asterisk in the namestring matches with any character string. By default the comparison test is not case sensitive.

`Name=*fish*`	Name must contain 'FISH'
`Name=01*`	Name must start with '01'
`Name=*5`	Name must end with '5'

Several namestrings can be used in one selection.

Name=fish*,*lake	Name must start with 'Fish' or end with 'Lake'

You can use characters * and ? which match with any character(s) in both E1FILTER.infile and fmk.name, but there are some differences in using these characters in these parameters:

- NAME accepts only *, but in INFILE you can use both * and ?
- In NAME you can use * anywhere, but in INFILE only in the end (everything after * in INFILE is disregarded)
- Handling characters * and ? in NAME is part of E1FILTER-program, but handling characters * and ? in INFILE is based on operating system services.

FMK.CONTAINS= string1,sting2,string3,…, selection by the content of the file

Selects the file, if the file contains any of the strings. By default the comparison test is not case sensitive

```
fmk.contains=pcx5 2.09
fmk.contains=map project abe,calibration data
```

This parameter can also be given in the command line between quotation marks.

```
E1FILTER "Invoicing summary"
E1FILTER "map project abe,calibration data"
```

Phase "Read input file block"

The default input-file format is simple text-file with carriage return + linefeed as line end indicator.

You can define another line end indicator with parameter *LineReadCmd*, and/or define fixed input line length with parameter *MaxLineLen* or binary input record structure using parameter *InLineMask*.

When input line is read it is formatted as defined in parameter *LineEdit*. Immediately after reading the input line E1FILTER performs character conversion, if one is defined using parameter *ConvFile*.

The exact flow is as follows:

if *InLineMask* is defined
    read the block as defined in *InLineMask*
else
    read as defined in *LineReadCmd* or *MaxLineLen* bytes, which ever is reached first
endif
format the block using *LineEdit*
make character conversion as defined in *ConvFile*

InLineMask=list of binary fields/block read descriptions

Parameter InLineMask can be used to describe the line structure by defining the field type of every column.

When InLineMask is defined, E1FILTER :

1. Converts all fields (column values) to text-strings using the field types
2. Assembles a new text line using the converted values and value of the parameter InSep as column separator
3. Uses the new assembled line as the input-line

Table of the recognized field types/block read commands.

Field value	Bytes	Description	Comment
BYTE	1	Byte	0-255
INT	2	Integer	Two's complement
USI	2	Unsigned integer	0-65535
LNG	4	Long integer	Two's complement
SGL	4	Single precision floating point	IEEE-standard
DBL	8	Double precision floating point	IEEE-standard
TEString	Variable	Reads up to and including 'String', but strips String away from the returned value. (T=To, E=Exclude)	String' may contain @nnn, where nnn is ASCII code of a character, e.g. TE@255A
TIString	Variable	As TEString, but the 'String' is included in the returned value (T=To, I=Include)	E.g. TI@

LINE	Variable	Up to and excluding CR+LF	Same as TE@13@10
COMMA	Variable	Up to and excluding comma (,)	Same as TE,
SPACE	Variable	Up to and excluding space	Same as TE@32
Number	Number	Fixed length text	

Examples:

```
InLineMask=10,10,15,20            Fixed length fields
InLineMask=LNG,DBL,DBL,15
InLineMask=15,16,DBL,TE@10
```

LineReadCmd=Block read description

This parameter defines the command to read one input line. The default value is TE@13@10, which means reading up to but not including CR+LF combination. See the table in parameter InLineMask to see other possible commands. E1FILTER reads an input-line as defined in LineReadCmd, unless the maximum line length (see parameter MaxLineLen) is reached before that.

Characters in the parameter-string can be given in form of @nnn, where nnn is character code.

MaxLineLen=Maximum length of the input line

Defines the maximum length of the input line. E1FILTER reads an input-line as defined in parameter LineReadCmd, unless the maximum line length is reached before that.

If the line length is not fixed and is maximum line length is not set by user, 2000 characters is used as default.

LineEdit=Line edit formatting instructions

A formatting instruction, which is applied to every input line. In this function O$ represents the edited line.

Default value: no formatting

In the example below every line is first trimmed (leading and trailing spaces removed), end then converted to Sentence Case:

```
LineEdit=SC:(tr:(O$))
```

ConvFile=name of the character conversion file

Name of a character conversion file. If the character conversion is defined, it is done to every line just before writing the line to the output-file. Default value: no conversion.

```
ConvFile=c:\util\AsciiOEM.cnv
```

Conversion file is a list of character pairs per line in the format “original character, new character, comment”, e.g.

```
A,X, A is converted to X
```

Characters can be given in form @nnn, where nnn is the ASCII-code of the character, e.g.

```
@146,@198,  Æ
@147,@244,  ô
@148,@246,  ö
@150,@251,  û
@151,@249,  ù
@152,@255,  ÿ
```

Phase "Build input line"

In this phase the E1FILTER builds up the input line and line items, to which you can refer using functions ILINE, I and II.

In addition to normal text- or binary column based file (default) it is possible to define two special file formats with parameter *InForm*; XML and STRUCTURED.

If the value of the parameter *InForm* if not defined (default) of is COLUMN, the input block if used as input line and E1FILTER splits the input line to items using the value of parameter *InSep* as item separator.

If the value of the parameter *InForm* is XML or STRUCTURED, the input line is assembled as described the description in parameter INFORM.

After this phase the value of the parameter *InForm,* that is, the original input format, has no effect on the process.

If any line item listed in parameter *CopyPrevCols* is empty, the value of the line item is copied value from the value of the same column in the previous line.

If the input-file contains a header, you should define the length of the header (number of lines) with parameter *InHeaderLen*. It is also possible to read the header information (column titles) from a separate file by giving the name of the file in parameter *HeaderFile*.

At the end of this phase the value of previous line used in functions J and JJ is saved.

Inform=Input-file format

Parameter INFORM tells the format of the input file:

COLUMN	Column based binary or text-file
XML	XML-file
Structured	Structured export format

The default value is COLUMN and therefore the parameter INFORM needs to be set only, if the input-format is XML or Structureds.

- INFORM=XML, eXtensive Markup Language

E1FILTER has a simple XML-parser, which reads XML-files and writes the information in simple character separated column format.

The output format can be one of the following formats (here 'n' is the level of the element in the data hierarchy):

```
1=header n,value n (default format)
2=header n,,,value n (where value n is in the n:th column)
3=header 1,header 2,...,header n,value
```

The format number is given after XML separated by comma, e.g.

`INFORM=XML`	Default value (1)
`INFORM=XML,3`	Output format is 3

E1FILTER'S XML-parser ignores the data schemas totally.

E1FILTER's XML-parser recognizes and handles properly program instructions, comments and cdata-blocks, but program instructions and comments are not transferred to the output file.

```
<!--     -->        Comments read properly, but not written
```

```
<?name pidata?>     Prog. inst. read properly, but not performed
<![CDATA[   ]]>     CDATA understood
```

When inform is XML, E1FILTER ignores the values of the parameters Oc and HeaderOut, because the values of these parameters are not applicable in XML-format.

Sample XML-input-file:

```
<gpx version="1.0" creator="GeoConv"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.topografix.com/GPX/1/0"
xsi:schemaLocation="http://www.topografix.com/GPX/1/0 gpx.xsd">
   <desc>From C:\GPSDATA\PCX5\SOUR\LANGVIK2.TRK</desc>
   <trk>
      <name>Route/Track 01</name>
      <desc>Route/Track 01</desc>
      <number>1</number>
      <trkseg>
         <trkpt lat="+060.000000" lon="+019.000000">
            <ele>0</ele>
            <time>2000-07-16T09:37:31Z</time>
            <geoidheight>0</geoidheight>
            <name>000001</name>
         </trkpt>
         <trkpt lat="+060.110292" lon="+024.565770">
            <ele>0</ele>
            <time>2000-07-16T09:37:31Z</time>
            <geoidheight>0</geoidheight>
            <name>000002</name>
         </trkpt>
```

Sample XML-output-file (output format 2=:

```
desc,,From C:\GPSDATA\PCX5\SOUR\LANGVIK2.TRK
name,,,Route/Track 01
desc,,,Route/Track 01
number,,,1
ele,,,,,0
time,,,,,2000-07-16T09:37:31Z

geoidheight,,,,,0
name,,,,,000001
ele,,,,          ,0
time,,,,,2000-07-16T09:37:31Z
geoidheight,,,,,
```

- INFORM=STRUCTURED, structured format export file
Parameter FILTERID.<field name>.<column number>=column number

If Inform is STRUCTURED, the input-file is processed as in structured format exported Structured-database file.

By default the ID-separator is colon and the record separator is form-feed (character code 12). If other values can be given in parameter INFORM after the text 'STRUCTURED' separated by commas, e.g.

`INFORM=STRUCTURED`	Default values
`INFORM=STRUCTURED,:,@12`	Default values
`INFORM=STRUCTURED,=,@9`	equal sign and tabulator

When Structured -input-format is used, the database field ID's must be associated with column numbers as fvalue-settings of value-set FILTERID in format:

fvalue FILTERID.<field name>.<column number>

Examples:

```
fvalue FILTERID.EntryDate=1
fvalue FILTERID.IssueCategory=2
fvalue FILTERID.Status=3
fvalue FILTERID.MBP=4
```

Optional formatting instruction and header-text can be given after column number. Column number, formatting instruction and header-text must be separated by double colon (::). If no header-text is given, the field-name is used instead.

fvalue FILTERID.<field name>.<column number>::<formatting>::<header-text>

```
fvalue FILTERID.Priority=5
fvalue FILTERID.PlannedStart=6
fvalue FILTERID.Owner=7::UC::Responsible
```

InSep=column separator of the input-file

This parameter sets the input-file column separator. Separator can be given as a character or in form of @nnn, where nnn is character code. Default value is tabulator character

`InSep=;`	separator is semicolon
`InSep=@9`	separator is tabulator character

- Insep=NONE

There is no input-separator, that is, every line has only one column

CopyPrevCols=Copy Previous Column values

Sometimes, when a column value in a file is blank, the desired function is, that the previous non-blank value of the same column is used instead. This can be achieved using parameter CopyPrevCols, where user can give a comma-separated list of columns, to which this parameter should be applied.

Examples:

```
CopyPrevCols=2
CopyPrevCols=2,7,13
```

A series of consecutive integers may be presented in form n-m, where n is the first and m is the last integer, e.g.:

`2,5,11-19,22,27` is same as `2,5,11,12,13,14,15,16,17,18,19,22,27`

InHeaderLen=number of the lines in the header

This parameter sets the number of lines in the beginning of the input-file belonging to the column header. If the number of headerlines is greater than one, only one headerline is printed to the output-file anyhow. In this case the titles of consecutive lines are appended.

```
InHeaderLen=
InHeaderLen=1
```

HeaderFile=name of separate Header file (column title file)

Column titles can be given in a separate Headerfile, where lines represent the column titles. If the input-file also contains Header-information (InHeaderLen>0), the column titles read from HeaderFile override the column titles read from the input-file.

`Invoice`	Title of the column 1 is read from line 1
`Project`	Title of the column 2 is read from line 2

```
Employee
```
etc.

Phase "Start/stop reading input file"

In this phase you can start and stop reading the input file based on the conditions defined in the formulas in parameters *StartReadForm* and *StopReadForm.*

If the value of the formula defined in the parameter *StartReadForm* is true, E1FILTER starts reading the input file starting from this line.

If the value of the formula defined in the parameter *StopReadForm* is true, E1FILTER stops reading the input file starting from this line.

If both *StartReadForm* and *StopReadForm* are true, E1FILTER stops reading the input file, because *StopReadForm* is checked last.

StartReadForm=Condition formula for starting the reading
StopReadForm Condition formula for stopping the reading

Formulas defining optional start and stop conditions for processing the input lines.

StartReadForm: if true, starts processing including current line
StopReadForm: if true, stops processing including current line

E.g.

```
StopReadForm=MATCH:(I$,?PAYMENT SUMMARY?)
```

Phase "Select input lines

E1FILTER selects (filters) the input lines as defined in parameters *Incl, Excl*.

Incln, where n=1-9, line selection conditions (Include)
Excln, where n=1-9, line exclusion conditions (Exclude)

Set of formulas to be used as input line selection criteria. An input-file line will be written to the output-file if and only if:

- All inclusion conditions are fulfilled
- None of the exclusion conditions is fulfilled

The maximum number of selections is 10, unless set differently with the parameter MaxScanCount=.

Phase "Build output line"

In this phase E1FILTER builds up the output line and line items, to which you can refer using functions OLINE, J and JJ.

E1FILTER assembles the output line as defined in parameter *Oc*, using the value of parameter *OutSep* as column (item) separator.

If the input line item contains output separator character defined in parameter *Outsep*, it is substituted by the character defined in parameter *RepSep*.

The output line times are formatted as defined in parameter *ItemEdit* prior to formatting the item as defined in parameter *Oc*'s formatting instruction.

Oc=Output columns

A comma separated list of columns to be written to the output-file. If the value of this parameter is empty, all columns will be written to the output-file.

Example:

```
Oc=1,15,4,13,2,5,18,19
oc=1,13-21
```

- Oc, additional parameters

Output column number can be followed by optional formatting instruction, optional column header and optional summarizing instruction. These are separated by double colon, e.g.

```
5::SST:(o$,1,9)::Customer
```

- Oc, formatting and header text

If the column header is given here, it overrides the column header read from the input file or from the optional Headerfile In the first example below the value of the sixth output column is the first four characters of the fifth input-column and the header of the sixth output column is set to Year.

```
Oc=1-5,5::SST:(o$,1,9)::Year

Oc=1,15,4,7::::Group,11,14,12::SST:(o$,1,9)::Phase
```

A formatting instruction may consist of several formatting commands, which are separated by spaces,. You can refer to the previous value (start value or the value after precious formatting) with function O$, e.g.

```
Oc=1-5,5::SST:(o$;2;7) uc:(o$);Year
```

- Oc, summarizing function

For description of summarizing action, see section **Phase "Summarize".**

- Oc, giving long list of consecutive integers

A series of consecutive integers may be presented in form n-m, where n is the first and m is the last integer. The range n-m can not be followed by formatting instruction, header or summarizing Parameter.

```
2,5,11-19,22,27     is same as 2,5,11,12,13,14,15,16,17,18,19,22,27
```

OutSep=column separator of the output-file

This parameter sets the output-file column separator. Separator can be given as a character or in form of @nnn, where nnn is character code. Dafault value is tabulator character

```
OutSep=;            separator is semicolon
OutSep=@9           separator is tabulator character
```

RepSep=Separator substituting OutSep-character in the input-item

If the input line item contains output separator character defined in parameter *Outsep*, it is substituted by the character defined in parameter *RepSep*. Characters in the parameter-string can be given in form of @nnn, where nnn is character code

ItemEdit=Item edit formatting instructions

A formatting instruction, which is applied to every item in the input line. In this function O$ represents the edited value.

In the example below every item is first trimmed .

```
ItemEdit= tr:(O$)
```

Phase "Select output lines

E1FILTER selects (filters) the output lines as defined in parameter *PrintOutCond*.

PrintOutCond=Condition formula to print out the line

A formula defining, whether the output line should be printed or not. This test is done just before the final output after all phases.

```
PrintOutCond$=NOT:(BL:(OLINE$))
```

Phase "Sort"

E1FILTER performs optional sort as defined in parameters *SortKey*n, where n is a number between 1 and 9.

SortKeyn=Sorting Parameters number n, n=1 to 9

The format of the sort key parameter is:

parameter string, key length, sort key formula

Where

1. *parameter string* = list of sorting parameters
2. *key length* = number of characters (bytes) in the key, default 30
3. *sortkey formula* = formula defining the sort key value

Possible sorting parameters in the sort key parameter string. (parameter 1)

Character	Short of	Description
D	Descend	Sort in Descending order
R	Right	Right justify the sort key value
N	Numeric	Sort according the Numeric value of the column

e.g.

```
FVALUE SortKey1=RN,,SST:(I$9;2;999)
```

Phase "Allocate values to calendar days"

Using parameters *ObjectForm, SplitCols, SplitEdit, SplitValueForm, AbsLoad, PeriodCol, PeriodForm, TimeScope, ExcludeFile* and *CalDateForm* you can define, that E1FILTER distributes (allocates, shares) the values of some columns in calendar days in period defined in another column.

General logic of Calendar allocation

For every input line E1FILTER defines one **Time period**. The **Time period** is either calculated by evaluating the formula in parameter *PeriodForm* or if *PeriodForm* is not defined, by reading the value in column *PeriodCol*

The goal is to allocate the values on columns listed in parameter *SplitCols* evenly on the **Time Period**. For that E1FILTER calculates the **Number of days** in the **Time Period.**

Then E1FILTER writes for one input line as many output lines as there are days in the time period (Output lines 1 to **Number of days,** one output line per day) using following rules per column:

- If column is *PeriodCol,* the particular date in the **Time period** is written in that column using date format described in parameter *CalDateForm.*
- If the column is listed in parameter *SplitCols,* the daily share of that column value (column value/**Number of days**) written in that column edited using editing instruction in parameter *SplitEdit*
- Otherwise the input column is written as is to the respective output column

The final result is, that the original input line defining a **Time Period** and total values for that **Time Period** in columns listed in parameter *SplitCols,* is split into daily lines containing the date and the daily shares of the total values.

E.g. Time Period is the forecast period and value to split is total forecast value. Then you want to split the total forecast line into daily lines for OLAP-reporting tool.

This default allocation method can be modified by using additional calendar allocation parameters.

PeriodForm=formula returning the time period

PeriodForm returns the time period for calendar allocation. If *PeriodForm* is not defined, E1FILTER uses the value of column number *PeriodCol* in the input line instead.

Allocation period can be given as day, month or year:

15.10.2005	15th October 2005
10.2005	October 2005
2005	Year 2005

Or as time range, where day, month and year can be mixed:

15.10.2004-2005	15.10.2004 – 31.12.2005
02.2005-11.2005	1.2.2005-30.11.2005

Or weekdays:

SAT	Saturdays
SUN	Sundays
Etc.	

Time periods can be combined using semicolons:

14.3.2004;18.3.2004-27.4.2004

E1FILTER understands several other ways of defining the time period, even quite complex, e.g.

- Every second month
- Every third Tuesday
- The second weekday of the month
- The first Monday after 10th day of the month
- Etc.

If you need these complex formats, which are not described here, please contact me

PeriodCol=number of column containing the time period
CalDateFormat=date format for calendar day output

If PeriodForm is not defined, E1FILTER uses the value of column number *PeriodCol* in the input line instead.

In output lines, the particular date in the **Time period** is written in *PeriodCol*-column using date format described in parameter *CalDateFormat.*

See more about available date formats in document Evaluator.doc, section 'Describing the input- and output-formats of date and time fields'

SplitCols=list of columns to be split into daily shares

If the column is listed in parameter *SplitCols,* the daily share of that column value (column value/ **Number of days**) written in the column.

ExcludeFile=name of the file containing dates to be excluded

File containing days to be excluded from the calendar allocation; no shared values will be allocated to the excluded dates. This file is a list of time period expressions as described under parameter *PeriodForm* e.g.:

```
SAT
SUN
1.1.2003
6.1.2003
24.12.2003-26.12.2003
1.1.2004
6.1.2004
```

IgnoreExcludeFileForm=Boolean formula: ExcludeFile should be ignored

Formula telling, if *ExcludeFile* should be ignored for this line.

E.g. You are using calendar allocation for resource allocations and for this resource weekends shouldn't be excluded.

ObjectForm=Formula returning the object (string)

Parameter *ExclusiveForm*, exclusive time allocation/reservation*,* is defined per Object, which is defined with parameter *ObjectForm*.

E.g. You are using calendar allocation for resource allocations and you want to make an exclusive reservation, e.g. course or holiday, for a resource,. Then you define the resource as an object (*ObjectForm* returns the resource) and set *ExclusiveForm* =Y or formula returning Y (Yes).

ExclusiveForm=Is an exclusive allocation (reservation)

Exclisive allocation means, that after this allocation nothing is allocated to same period of time for same **object**. (see parameter *ObjectForm).*

Exclusive allocation could be used e.g. in resource planning to indicate vacation time; you can allocate any other task at the same time.

If you want make sure, that this exclusion affects all other allocation for the same object, you have to sort the input file so, that exclusive reservations come first.

TimeScope=Time period limitation for calendar allocation

You may limit the calendar allocation to a predefined period. Then all allocations outside the *TimeScope* are ignored. For time period expressions, see parameter *PeriodForm*

E.g.

```
TimeScope=2007-2009
```

SplitValuesForm=Formula telling, if split values are copied 1:1 to dates

Normally the split values (values in the columns listed in *SplitCols*) are distributed evenly to the days in the allocation period. If *SplitValuesForm*=Y, the original values are copied 1:1 to the days in the allocation period.

E.g. You are using calendar allocation for resource allocations and you define the allocation ratio in a column and want that allocation ratio to be copied to all allocated days as is..

MaxAllocCountForm=Formula returning maximum number of allocated days

Gives the desired (=maximum) number of days, on which the splits is allocated (=days*load). E1FILTER starts counting from the period start and stops, when the maximum number of days is full or the period end is encountered, which ever comes first. If the period has only end date, the counting starts from the end.

E.g. You are using calendar allocation for resource allocations and you want to make reservation for fixed number of days.

Phase "Summarize"

Instead of printing out all lines you can print summary lines. The optional summary lines are defined in the parameter *SumKeyForm*, which defines the summary groups and appropriate sub-parameters in parameter *Oc*, which defile the summary value calculation.

Because the summarizing information is printed out every time the key value changes, you might want to sort the input file in summarizing key order. This however is not mandatory; if the same key appears two or more times in the input data, you get several sums of the same key, but the total sum is correct.

SumKeyForm=Summary key calculation formula

When a new line is read from the input file, E1FILTER calculates the summary key for the new line by evaluating the formula on the parameter SumKeyForm.

Then E1FILTER compares the summary keys of the new line and the previous line.

If the keys equal, E1FILTER accumulates the summary values using summary functions defined per column in parameter *oc*. and does **not** print out the line.

If the keys differ, E1FILTER prints out the summary line using following rules:

- if there is a summary function defined for the column, the summary value is printed
- otherwise, if within the summary group all the item values in a column are same. the value is regarded as constant and printed out
- otherwise an empty column is printed

Summary function (in parameter oc)

A function defining the summary value of the column. Function O$ used in summary function refers to the previous accumulated summary value.

E.g.

`o$+I$6` The summary value is the sum of value in the column 6

Phase "Create output columns"

The final phase is writing the output file. The name of the output file is defined in parameter *OutFile*.

The name of the output file (*OutFile*) can be fixed it can contain variable parts indicated by formulas enclosed in brackets. Using variable parts you can make the name of the output file(s) dependent on the content of the input file.

If the name of the output-file is not complete, that is, if e.g. the base or the extension of the name is missing, E1FILTER takes the missing parts of the input-file-name.

By default E1FILTER removes the output file before writing the first line in the file. E1FILTER however retains the existing content and appends the new lines to the end of the existing file, if

- parameter *Append* is set (=Y) or
- the name of the output file contains variable parts (in brackets)

N/B: However, if E1FILTER doesn't write anything to the output-file, the old file will remain intact.

Parameter *HeaderOut* defines, if the header should be written in the output file.

By default E1FILTER uses carriage return + linefeed (CR+LF) as the line end indicator in the output file, but this can be changed using parameter *OutLineEndStr.*

OutFile=name of the output-file

Parameter *OutFile* defines the name of the output-file.

Example 1, complete input-file-name

```
OutFile=d:\temp\issues.txt
```

Example 2, part of the name of the output-file is taken from the name of the input-file.

Input-file	Output-file	Completed output-file-name
`c:\invoice.csv`	`d:\.txt`	`d:\invoice.txt`
`\project.csv`	`d:\expense`	`d:\expence.csv`

Append=Append to the output file

If the output-file is defined with parameter OutFile and a file with the same name already exists, it will be overwritten (deleted), before E1FILTER writes to it.

With parameter NameCol you can define, that the new lines are appended to the end of the existing file in both cases.

HeaderOut=Y/N

Defines whether the Header is printed to the output-file or not.

OutLineEndStr=character string

This parameter defines the character string used as line end string, when writing output lines. The default value is @13@10 (CR+LF). Characters in the string can be given in form of @nnn, where nnn is character code.

`OutLineEndStr=`	Line end string is the default value is CR+LF
`OutLineEndStr=//@20`	Line end string is two slashes + character 20

Phase "Format output"

LineTemplate=Template for line output
HeaderTemplate=Template for header block output
FooterTemplate=Template for footer block output

If the output template parameters above are not defined, E1FILTER prints

- column names as header
- output columns as defined in parameter *Oc* and separated by *OutSep* as output lines
- no footer

With template parameters you can defines alternative output layouts. Templates are simple text blocks, but you can insert variable parts (function blocks) enclosed in brackets within the text

e.g.
e1filter.linetemplate=[i:3] temp [@9]dev. point [dr:(x;i:2][@9][i:8] date [dat:(i:1;DD.MM;]

You can define even several lines long template by defining template-text-files and using marker file: in template parameter. If you e.g. prepare file "header.txt" as header template, you define

e1filter.headertemplate=file:header.txt

If you use parameter *LineTemplate*, you may omit parameter *oc*, if you don't want to teh editing power of parameter *oc* (to edit columns for use of *LineTemplate*)

Application functions of E1FILTER

E1FILTER is based on Formula Evaluator. In addition to Formula Evaluator's standard functions E1FILTER has a set of it's own application functions , which are listed in the table below.

The only difference between the usage of standard functions and application functions is, that applications functions are normally indicated by a dollar sign (e.g. LINE$) while standard functions are always indicated by a colon (e.g. DAT:). It is how ever possible to use colon also with application functions.

Standard and application functions can be freely mixed. e.g. (SST:(OLINE$;7;9).

Function	Description	Syntax
ILINE	Input line	ILINE$
OLINE	Output line	OLINE$
I	Input line item	I$num or I$Column name
J	Output line item	J$num or J$Column name
II	Previous input line item	II$num orII$Column name
JJ	Previous output line item	JJ$num or JJ$Column name
INCNT	Input line count	INCNT$
OUTCNT	Output line count	OUTCNT$
O	Object	O$
P	Parameter	P$ or P$parameter number
FILE	Name of the input file	FILE$

Hints, tricks, warnings and examples

Using filenames longer then eight characters

Because E1FILTER is a DOS-program, it doesn't understand file- or pathnames longer than 8 characters. However, in the DOS-command prompts of the latest Windows versions you can use longer filenames. Then you handle longer filenames in batch files using copy or rename command before calling E1FILTER, e.g.

```
call FILTEINI.bat
copy TransactionsData2003A.txt Data2003.txt
fvalue E1FILTER.infile=Data2003.txt
fvalue E1FILTER.outfile=…
```

Using full path names

All examples above assume, that the converted files are located in the working/default-directory. If you have a lot of files from various sources, you don't want to store them all in one directory. Then you have to include the full pathname in the filename, e.g.:

```
call FILTEINI.bat
fvalue E1FILTER.infile=c:\trasdat\2003\Incoice\Data2003.txt
fvalue E1FILTER.outfile=c:\transdat\report\Report2.txt
```

Creating a log-file

If you create long and/or complex batch-runs, you might want to create a log-file. E1FILTER writes all error messages in the standard output, and therefore you can create a log-file by redirecting the standard output to log-file, e.g.:

```
echo Creating invoice-files >\deleteme.log
call FILTEINI.bat
fvalue E1FILTER.infile=c:\transdat\2003\trans\*.txt
fvalue E1FILTER.outfile=c:\transdat\2003\Invoices\.txt
fvalue E1FILTER.NameCol=8R
E1FILTER >>\deleteme.log
```

Displaying the results

If you want to have a look at the results during or at the end of the batch file, you can use program show.exe, e.g.:

```
....
E1FILTER >>\deleteme.log
show c:\transdat\2003\Invoices\output.txt
show \deleteme.log
```

You can exit show with Esc-key. Pressing Alt+5 changes to 50 rows/screen display and Alt+2 again 25 rows/screen display.

Using Fvalue-values in the command-line

If you write a Fvalue-instruction closed in double brackets in the command-line of programs FVALUE, E1FILTER or SHOW, it will be replaced by its value. This can be used e.g. for

Using own favorites saved in a separate file:

```
Fvalue E1FILTER.Infile=[[Myvalues.CurrentTransactionFile]]
```

Writing long value in parts:

```
fvalue E1FILTER.oc=1,2-5
```

```
fvalue E1FILTER.oc=[[E1FILTER.oc]],4;fv.project.txt./3/@9/6;PM
fvalue E1FILTER.oc=[[E1FILTER.oc]],7-13,19
E1FILTER.oc=[[E1FILTER.oc]],4;fv.project.txt./3/@9/4;Project desc
E1FILTER.oc=[[E1FILTER.oc]],4;fv.project.txt./3/@9/7;Cust no
```

Avoiding writing same values many times:

```
Show [[E1FILTER.outfile]]
```

Setting a very long parameter value, which difficult to enter in the command line and is therefore created with a text-editor or similar in advance:

```
Fvalue E1FILTER.oc=[[myvalues.ColumnsForAccountInfo]]
```